object.py

```python
from utils import *
from shape3d import Vertex, Shape2D, Shape3D

class Object:
    def __init__(self,
            _position : Vertex = Vertex(0, 0, 0),
            _face : Vertex = Vertex(0, 0, 1),
            _shape : Shape2D | Shape3D = None,
            _edge : float = 1
            ) -> None:
        self.position : Vertex = _position
        self.face : Vertex = _face
        self.translation : list[float] = None
        self.rotation : tuple[float, str] = None
        self.shape : Shape2D | Shape3D = _shape
        self.edge : float = _edge

    def __repr__(self) -> str:
        return f"POS:  {self.position}\npos:  {self.shape.position}\nFACE: {self.face}\nT: {as_colored(self.translation, fg=FOREGROUND_COLORS_STRONG['yellow'])}\nROT: {as_colored(self.rotation, fg=FOREGROUND_COLORS_STRONG['red'])}\n{self.shape.position - self.shape.vertices[0]}"

    def push(self) -> None:
        """
        Give a random uniform vector of traslation
        """
        self.translation = [
            uniform(*UNIFORM_DIRECTION_AREA),
            uniform(*UNIFORM_DIRECTION_AREA),
            uniform(*UNIFORM_DIRECTION_AREA)
        ]

    def add_translation(self, _vector : list[float]) -> None:
        self.translation = _vector

    def add_rotation(self, _angle : float, _axis : str) -> None:
        self.rotation = (_angle, _axis)

    def draw_vector(self, _vector):
        glLineWidth(2)
        glBegin(GL_LINES)
        glColor3f(*COLORS['cyan'])
        glVertex3f(self.position.x, self.position.y, self.position.z)
        glVertex3f(self.position.x + _vector.x, self.position.y + _vector.y, self.position.z + _vector.z)
        glEnd()
        glLineWidth(1)

    def draw_y_axis(self):
        glLineWidth(2)
        glBegin(GL_LINES)
        glColor3f(*COLORS['red'])
        glVertex3f(self.position.x, self.position.y+1, self.position.z)
        glVertex3f(self.position.x, self.position.y-1, self.position.z)
        glEnd()
        glLineWidth(1)

    def draw_center(self):
        glPointSize(8)
        glBegin(GL_POINTS)
        glColor3f(*COLORS['white'])
        glVertex3f(self.position.x, self.position.y, self.position.z)
        glEnd()
        glPointSize(1)

        glPointSize(8)
        glBegin(GL_POINTS)
        glColor3f(*COLORS['cyan'])
        glVertex3f(self.position.x + self.shape.position.x, self.position.y + self.shape.position.y, self.position.z + self.shape.position.z)
        glEnd()
        glPointSize(1)

    def draw(self, _draw_edges : bool = False, _draw_vertices : bool = False) -> None:

        self.shape.draw(self.position, _draw_edges, _draw_vertices)

    def translate(self) -> None:
        if self.translation:
            self.position.translate(self.translation)


    def rotate(self) -> None:
        """
        Rotation in relation to Object Origin (Rotating only the object position and face Vector).
        """
        if self.rotation:
            if len(self.rotation) == 2:
                self.face.rotate(Vertex(0, 0, 0), *self.rotation)
                # self.position.rotate(self.position, *self.rotation)
                self.shape.rotate(Vertex(0, 0, 0), *self.rotation)
            elif len(self.rotation) == 3:
                self.face.rotate_any(Vertex(0, 0, 0), *self.rotation)
                # self.position.rotate_any(self.position, *self.rotation)
                self.shape.rotate_any(Vertex(0, 0, 0), *self.rotation)


    def spin(self):
        self.rotation = [
            uniform(*UNIFORM_DIRECTION_AREA),
            uniform(*UNIFORM_DIRECTION_AREA),
            uniform(*UNIFORM_DIRECTION_AREA)
        ]

    def collide(self):
        if self.translation:
            collision = [1, 1, 1]
            if self.position.x > BORDER_COLLITION_CAP or self.position.x < -BORDER_COLLITION_CAP:
                collision[0] = -1
            if self.position.y > BORDER_COLLITION_CAP or self.position.y < -BORDER_COLLITION_CAP:
                collision[1] = -1
            if self.position.z > BORDER_COLLITION_CAP or self.position.z < -BORDER_COLLITION_CAP:
                collision[2] = -1

            self.translation = [
                self.translation[0] * collision[0],
                self.translation[1] * collision[1],
                self.translation[2] * collision[2]]


if __name__ == '__main__':
    pass

```